An Introduction to the Conjugate Gradient Method Without the Agonizing Pain

Edition $1\frac{1}{4}$

Jonathan Richard Shewchuk

August 4, 1994

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The Conjugate Gradient Method is the most prominent iterative method for solving sparse systems of linear equations. Unfortunately, many textbook treatments of the topic are written with neither illustrations nor intuition, and their victims can be found to this day babbling senselessly in the corners of dusty libraries. For this reason, a deep, geometric understanding of the method has been reserved for the elite brilliant few who have painstakingly decoded the mumblings of their forebears. Nevertheless, the Conjugate Gradient Method is a composite of simple, elegant ideas that almost anyone can understand. Of course, a reader as intelligent as yourself will learn them almost effortlessly.

The idea of quadratic forms is introduced and used to derive the methods of Steepest Descent, Conjugate Directions, and Conjugate Gradients. Eigenvectors are explained and used to examine the convergence of the Jacobi Method, Steepest Descent, and Conjugate Gradients. Other topics include preconditioning and the nonlinear Conjugate Gradient Method. I have taken pains to make this article easy to read. Sixty-six illustrations are provided. Dense prose is avoided. Concepts are explained in several different ways. Most equations are coupled with an intuitive interpretation.

Supported in part by the Natural Sciences and Engineering Research Council of Canada under a 1967 Science and Engineering Scholarship and by the National Science Foundation under Grant ASC-9318163. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either express or implied, of NSERC, NSF, or the U.S. Government.

Keywords: conjugate gradient method, preconditioning, convergence analysis, agonizing pain

Contents

1. Introduction **1**

2. Notation **1**

3. The Quadratic Form **2**

4. The Method of Steepest Descent **6**

5. Thinking with Eigenvectors and Eigenvalues **9**
5.1. Eigen do it if I try . . . 9
5.2. Jacobi iterations . . . 11
5.3. A Concrete Example . . . 12

6. Convergence Analysis of Steepest Descent **13**
6.1. Instant Results . . . 13
6.2. General Convergence . . . 17

7. The Method of Conjugate Directions **21**
7.1. Conjugacy . . . 21
7.2. Gram-Schmidt Conjugation . . . 25
7.3. Optimality of the Error Term . . . 26

8. The Method of Conjugate Gradients **30**

9. Convergence Analysis of Conjugate Gradients **32**
9.1. Picking Perfect Polynomials . . . 33
9.2. Chebyshev Polynomials . . . 35

10. Complexity **37**

11. Starting and Stopping **38**
11.1. Starting . . . 38
11.2. Stopping . . . 38

12. Preconditioning **39**

13. Conjugate Gradients on the Normal Equations **41**

14. The Nonlinear Conjugate Gradient Method **42**
14.1. Outline of the Nonlinear Conjugate Gradient Method . . . 42
14.2. General Line Search . . . 43
14.3. Preconditioning . . . 47

A Notes **48**

B Canned Algorithms **49**
B1. Steepest Descent . . . 49
B2. Conjugate Gradients . . . 50
B3. Preconditioned Conjugate Gradients . . . 51

B4. Nonlinear Conjugate Gradients with Newton-Raphson and Fletcher-Reeves 52
B5. Preconditioned Nonlinear Conjugate Gradients with Secant and Polak-Ribière 53

C Ugly Proofs **54**
C1. The Solution to $Ax = b$ Minimizes the Quadratic Form 54
C2. A Symmetric Matrix Has n Orthogonal Eigenvectors. 54
C3. Optimality of Chebyshev Polynomials 55

D Homework **56**

About this Article

An electronic copy of this article is available by anonymous FTP to `WARP.CS.CMU.EDU` (IP address 128.2.209.103) under the filename `quake-papers/painless-conjugate-gradient.ps`. A PostScript file containing full-page copies of the figures herein, suitable for transparencies, is available as `quake-papers/painless-conjugate-gradient-pics.ps`. Most of the illustrations were created using Mathematica.

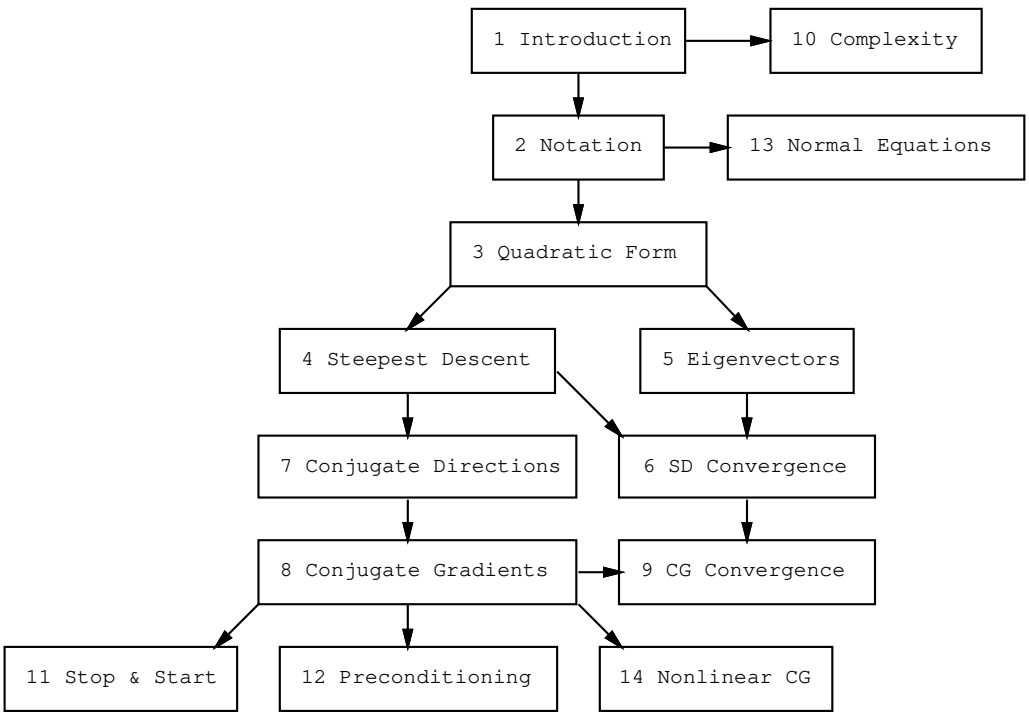
©1994 by Jonathan Richard Shewchuk. This article may be freely duplicated and distributed so long as no consideration is received in return, and this copyright notice remains intact.

This guide was created to help students learn Conjugate Gradient Methods as easily as possible. Please mail me (`jrs@cs.cmu.edu`) comments, corrections, and any intuitions I might have missed; some of these will be incorporated into a second edition. I am particularly interested in hearing about use of this guide for classroom teaching.

For those who wish to learn more about iterative methods, I recommend William L. Briggs' "A Multigrid Tutorial" [2], one of the best-written mathematical books I have read.

Special thanks to Omar Ghattas, who taught me much of what I know about numerical methods, and provided me with extensive comments on the first draft of this article. Thanks also to James Epperson, David O'Hallaron, James Stichnoth, Nick Trefethen, and Daniel Tunkelang for their comments.

To help you skip chapters, here's a dependence graph of the sections:



This article is dedicated to every mathematician who uses figures as abundantly as I have herein.

1. Introduction

When I decided to learn the Conjugate Gradient Method (henceforth, CG), I read four different descriptions, which I shall politely not identify. I understood none of them. Most of them simply wrote down the method, then proved its properties without any intuitive explanation or hint of how anybody might have invented CG in the first place. This article was born of my frustration, with the wish that future students of CG will learn a rich and elegant algorithm, rather than a confusing mass of equations.

CG is the most popular iterative method for solving large systems of linear equations. CG is effective for systems of the form

$$Ax = b \tag{1}$$

where x is an unknown vector, b is a known vector, and A is a known, square, symmetric, positive-definite (or positive-indefinite) matrix. (Don't worry if you've forgotten what "positive-definite" means; we shall review it.) These systems arise in many important settings, such as finite difference and finite element methods for solving partial differential equations, structural analysis, circuit analysis, and math homework.

Iterative methods like CG are suited for use with sparse matrices. If A is dense, your best course of action is probably to factor A and solve the equation by backsubstitution. The time spent factoring a dense A is roughly equivalent to the time spent solving the system iteratively; and once A is factored, the system can be backsolved quickly for multiple values of b. Compare this dense matrix with a sparse matrix of larger size that fills the same amount of memory. The triangular factors of a sparse A usually have many more nonzero elements than A itself. Factoring may be impossible due to limited memory, and will be time-consuming as well; even the backsolving step may be slower than iterative solution. On the other hand, most iterative methods are memory-efficient and run quickly with sparse matrices.

I assume that you have taken a first course in linear algebra, and that you have a solid understanding of matrix multiplication and linear independence, although you probably don't remember what those eigenthingies were all about. From this foundation, I shall build the edifice of CG as clearly as I can.

2. Notation

Let us begin with a few definitions and notes on notation.

With a few exceptions, I shall use capital letters to denote matrices, lower case letters to denote vectors, and Greek letters to denote scalars. A is an $n \times n$ matrix, and x and b are vectors — that is, $n \times 1$ matrices. Written out fully, Equation 1 is

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & & A_{2n} \\ \vdots & & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

The *inner product* of two vectors is written $x^T y$, and represents the scalar sum $\sum_{i=1}^{n} x_i y_i$. Note that $x^T y = y^T x$. If x and y are orthogonal, then $x^T y = 0$. In general, expressions that reduce to 1×1 matrices, such as $x^T y$ and $x^T A x$, are treated as scalar values.

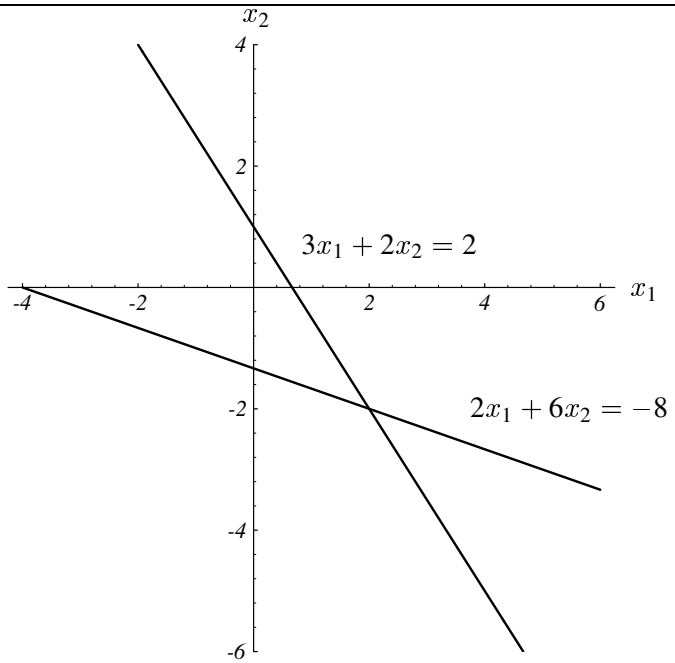


Figure 1: Sample two-dimensional linear system. The solution lies at the intersection of the lines.

A matrix A is *positive-definite* if, for every nonzero vector x,

$$x^T Ax > 0. \tag{2}$$

This may mean little to you, but don't feel bad; it's not a very intuitive idea, and it's hard to imagine how a matrix that is positive-definite might look differently from one that isn't. We will get a feeling for what positive-definiteness is about when we see how it affects the shape of quadratic forms.

Finally, don't forget the important basic identities $(AB)^T = B^T A^T$ and $(AB)^{-1} = B^{-1}A^{-1}$.

3. The Quadratic Form

A *quadratic form* is simply a scalar, quadratic function of a vector with the form

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c \tag{3}$$

where A is a matrix, x and b are vectors, and c is a scalar constant. I shall show shortly that if A is symmetric and positive-definite, $f(x)$ is minimized by the solution to $Ax = b$.

Throughout this paper, I will demonstrate ideas with the simple sample problem

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \qquad c = 0. \tag{4}$$

The system $Ax = b$ is illustrated in Figure 1. In general, the solution x lies at the intersection point of n hyperplanes, each having dimension $n - 1$. For this problem, the solution is $x = [2, -2]^T$. The corresponding quadratic form $f(x)$ appears in Figure 2. A contour plot of $f(x)$ is illustrated in Figure 3.

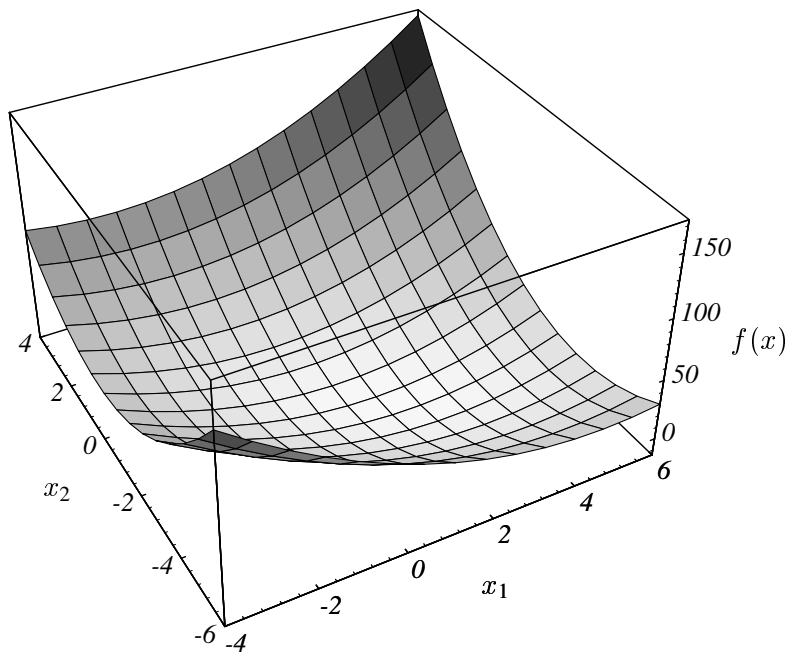


Figure 2: Graph of a quadratic form $f(x)$. The minimum point of this surface is the solution to $Ax = b$.

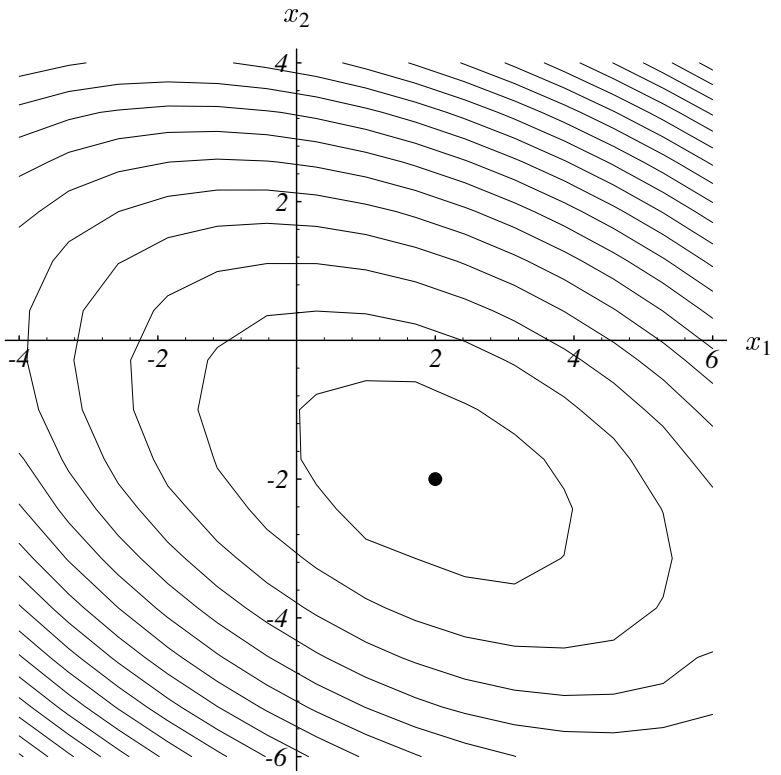


Figure 3: Contours of the quadratic form. Each ellipsoidal curve has constant $f(x)$.

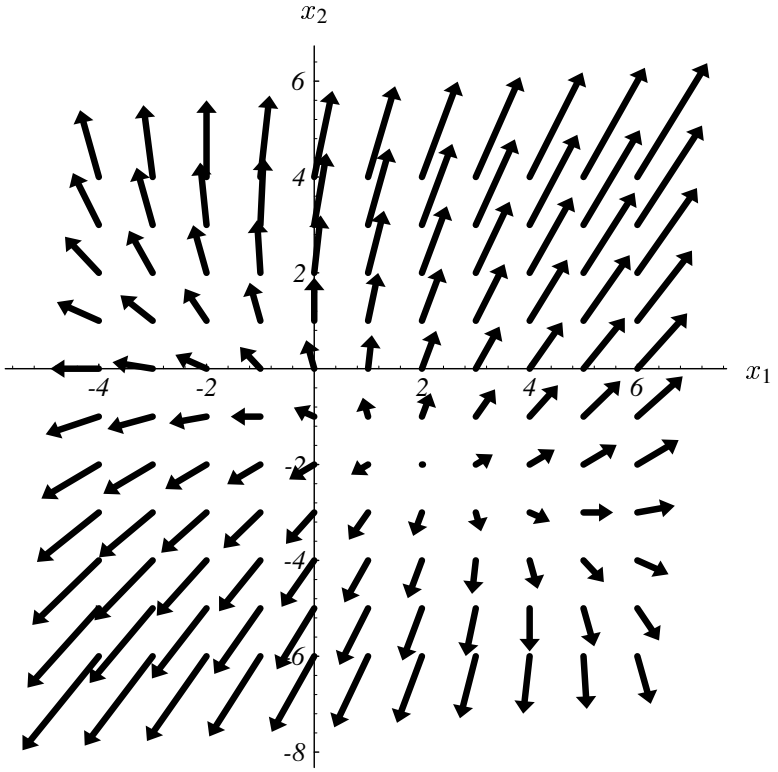


Figure 4: Gradient $f'(x)$ of the quadratic form. For every x, the gradient points in the direction of steepest increase of $f(x)$, and is orthogonal to the contour lines.

Because A is positive-definite, the surface defined by $f(x)$ is shaped like a paraboloid bowl. (I'll have more to say about this in a moment.)

The *gradient* of a quadratic form is defined to be

$$f'(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix}. \tag{5}$$

The gradient is a vector field that, for a given point x, points in the direction of greatest increase of $f(x)$. Figure 4 illustrates the gradient vectors for Equation 3 with the constants given in Equation 4. At the bottom of the paraboloid bowl, the gradient is zero. One can minimize $f(x)$ by setting $f'(x)$ equal to zero.

With a little bit of tedious math, one can apply Equation 5 to Equation 3, and derive

$$f'(x) = \frac{1}{2} A^T x + \frac{1}{2} A x - b. \tag{6}$$

If A is symmetric, this equation reduces to

$$f'(x) = Ax - b. \tag{7}$$

Setting the gradient to zero, we obtain Equation 1, the linear system we wish to solve. Therefore, the solution to $Ax = b$ is a critical point of $f(x)$. If A is positive-definite as well as symmetric, then this

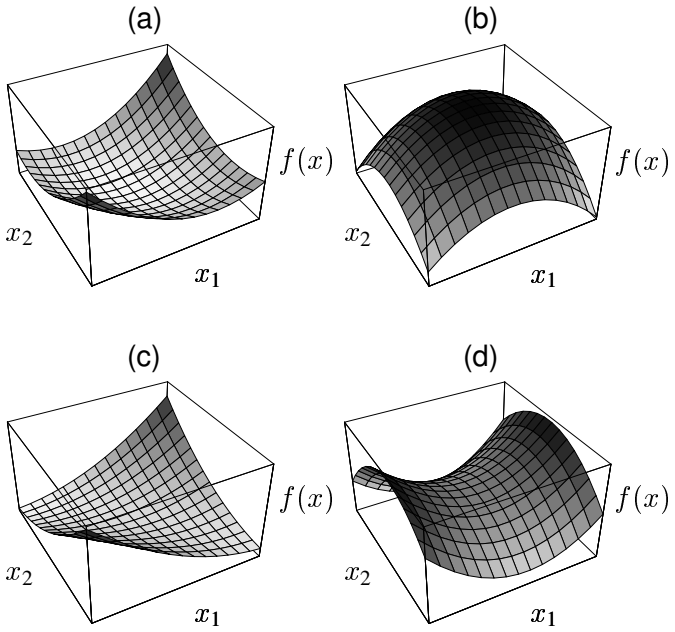


Figure 5: (a) Quadratic form for a positive-definite matrix. (b) For a negative-definite matrix. (c) For a singular (and positive-indefinite) matrix. A line that runs through the bottom of the valley is the set of solutions. (d) For an indefinite matrix. Because the solution is a saddle point, Steepest Descent and CG will not work. In three dimensions or higher, a singular matrix can also have a saddle.

solution is a minimum of $f(x)$, so $Ax = b$ can be solved by finding an x that minimizes $f(x)$. (If A is not symmetric, then Equation 6 hints that CG will find a solution to the system $\frac{1}{2}(A^T + A)x = b$. Note that $\frac{1}{2}(A^T + A)$ is symmetric.)

Why do symmetric positive-definite matrices have this nice property? Consider the relationship between f at some arbitrary point p and at the solution point $x = A^{-1}b$. From Equation 3 one can show (Appendix C1) that if A is symmetric (be it positive-definite or not),

$$f(p) = f(x) + \frac{1}{2}(p - x)^T A(p - x). \tag{8}$$

If A is positive-definite as well, then by Inequality 2, the latter term is positive for all $p \neq x$. It follows that x is a global minimum of f.

The fact that $f(x)$ is a paraboloid is our best intuition of what it means for a matrix to be positive-definite. If A is not positive-definite, there are several other possibilities. A could be negative-definite — the result of negating a positive-definite matrix (see Figure 2, but hold it upside-down). A might be singular, in which case no solution is unique; the set of solutions is a line or hyperplane having a uniform value for f. If A is none of the above, then x is a saddle point, and techniques like Steepest Descent and CG will likely fail. Figure 5 demonstrates the possibilities. The values of b and c determine where the minimum point of the paraboloid lies, but do not affect the paraboloid's shape.

Why go to the trouble of converting the linear system into a tougher-looking problem? The methods under study — Steepest Descent and CG — were developed and are intuitively understood in terms of minimization problems like Figure 2, not in terms of intersecting hyperplanes such as Figure 1.

4. The Method of Steepest Descent

In the method of Steepest Descent, we start at an arbitrary point $x_{(0)}$ and slide down to the bottom of the paraboloid. We take a series of steps $x_{(1)}, x_{(2)}, \ldots$ until we are satisfied that we are close enough to the solution x.

When we take a step, we choose the direction in which f decreases most quickly, which is the direction opposite $f'(x_{(i)})$. According to Equation 7, this direction is $-f'(x_{(i)}) = b - Ax_{(i)}$.

Allow me to introduce a few definitions, which you should memorize. The *error* $e_{(i)} = x_{(i)} - x$ is a vector that indicates how far we are from the solution. The *residual* $r_{(i)} = b - Ax_{(i)}$ indicates how far we are from the correct value of b. It is easy to see that $r_{(i)} = -Ae_{(i)}$, and you should think of the residual as being the error transformed by A into the same space as b. More importantly, $r_{(i)} = -f'(x_{(i)})$, and you should also think of the residual as the direction of steepest descent. For nonlinear problems, discussed in Section 14, only the latter definition applies. So remember, whenever you read "residual", think "direction of steepest descent."

Suppose we start at $x_{(0)} = [-2, -2]^T$. Our first step, along the direction of steepest descent, will fall somewhere on the solid line in Figure 6(a). In other words, we will choose a point

$$x_{(1)} = x_{(0)} + \alpha r_{(0)}. \tag{9}$$

The question is, how big a step should we take?

A *line search* is a procedure that chooses α to minimize f along a line. Figure 6(b) illustrates this task: we are restricted to choosing a point on the intersection of the vertical plane and the paraboloid. Figure 6(c) is the parabola defined by the intersection of these surfaces. What is the value of α at the base of the parabola?

From basic calculus, α minimizes f when the *directional derivative* $\frac{d}{d\alpha} f(x_{(1)})$ is equal to zero. By the chain rule, $\frac{d}{d\alpha} f(x_{(1)}) = f'(x_{(1)})^T \frac{d}{d\alpha} x_{(1)} = f'(x_{(1)})^T r_{(0)}$. Setting this expression to zero, we find that α should be chosen so that $r_{(0)}$ and $f'(x_{(1)})$ are orthogonal (see Figure 6(d)).

There is an intuitive reason why we should expect these vectors to be orthogonal at the minimum. Figure 7 shows the gradient vectors at various points along the search line. The slope of the parabola (Figure 6(c)) at any point is equal to the magnitude of the projection of the gradient onto the line (Figure 7, dotted arrows). These projections represent the rate of increase of f as one traverses the search line. f is minimized where the projection is zero — where the gradient is orthogonal to the search line.

To determine α, note that $f'(x_{(1)}) = -r_{(1)}$, and we have

$$\begin{aligned}
r_{(1)}^T r_{(0)} &= 0 \\
(b - Ax_{(1)})^T r_{(0)} &= 0 \\
(b - A(x_{(0)} + \alpha r_{(0)}))^T r_{(0)} &= 0 \\
(b - Ax_{(0)})^T r_{(0)} - \alpha (Ar_{(0)})^T r_{(0)} &= 0 \\
(b - Ax_{(0)})^T r_{(0)} &= \alpha (Ar_{(0)})^T r_{(0)} \\
r_{(0)}^T r_{(0)} &= \alpha r_{(0)}^T (Ar_{(0)}) \\
\alpha &= \frac{r_{(0)}^T r_{(0)}}{r_{(0)}^T Ar_{(0)}}.
\end{aligned}$$

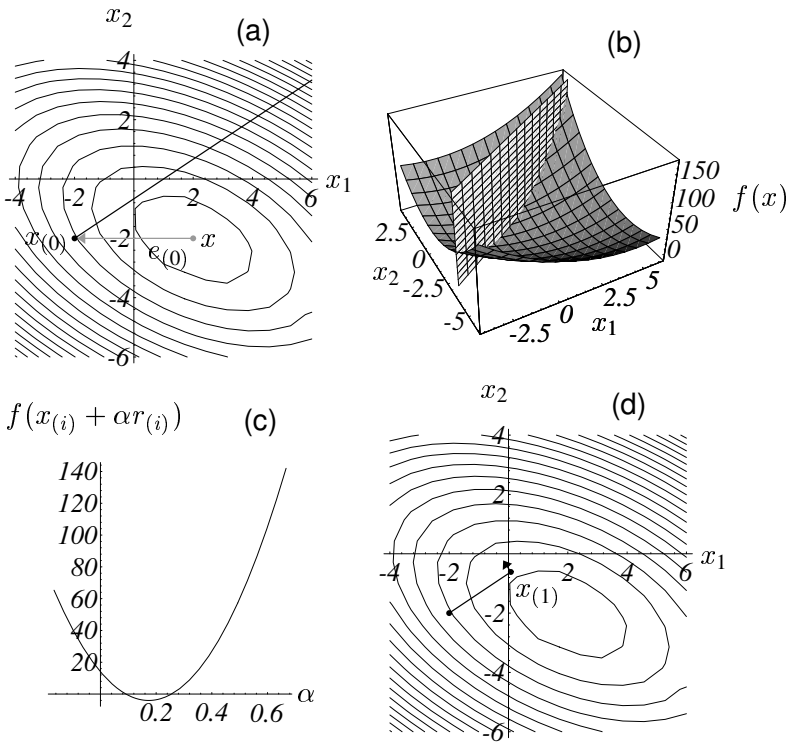


Figure 6: The method of Steepest Descent. (a) Starting at $[-2, -2]^T$, take a step in the direction of steepest descent of f. (b) Find the point on the intersection of these two surfaces that minimizes f. (c) This parabola is the intersection of surfaces. The bottommost point is our target. (d) The gradient at the bottommost point is orthogonal to the gradient of the previous step.

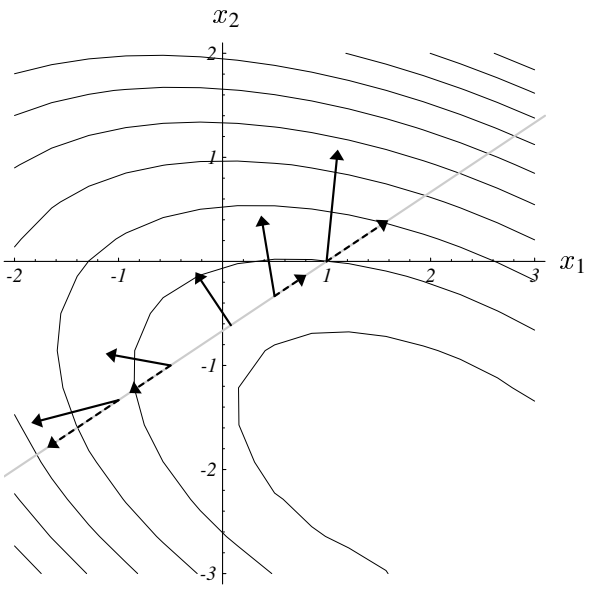


Figure 7: The gradient f' is shown at several locations along the search line (solid arrows). Each gradient's projection onto the line is also shown (dotted arrows). The gradient vectors represent the direction of steepest increase of f, and the projections represent the rate of increase as one traverses the search line. On the search line, f is minimized where the gradient is orthogonal to the search line.

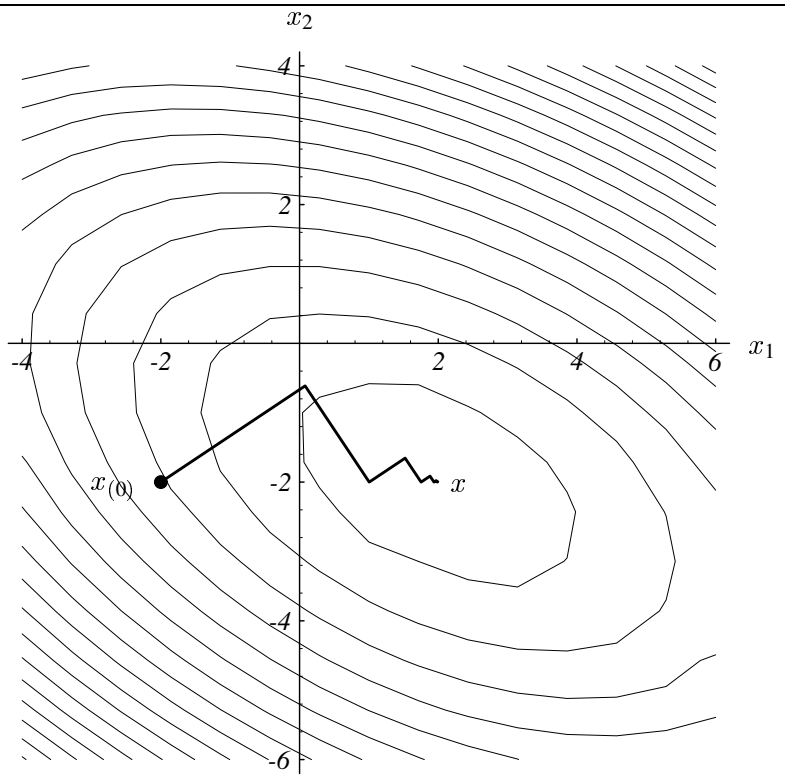


Figure 8: Here, the method of Steepest Descent starts at $[-2, -2]^T$ and converges at $[2, -2]^T$.

Putting it all together, the method of Steepest Descent is:

$$r_{(i)} = b - Ax_{(i)}, \tag{10}$$

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}}, \tag{11}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}. \tag{12}$$

The example is run until it converges in Figure 8. Note the zigzag path, which appears because each gradient is orthogonal to the previous gradient.

The algorithm, as written above, requires two matrix-vector multiplications per iteration. The computational cost of Steepest Descent is dominated by matrix-vector products; fortunately, one can be eliminated. By premultiplying both sides of Equation 12 by $-A$ and adding b, we have

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A r_{(i)}. \tag{13}$$

Although Equation 10 is still needed to compute $r_{(0)}$, Equation 13 can be used for every iteration thereafter. The product Ar, which occurs in both Equations 11 and 13, need only be computed once. The disadvantage of using this recurrence is that the sequence defined by Equation 13 is generated without any feedback from the value of $x_{(i)}$, so that accumulation of floating point roundoff error may cause $x_{(i)}$ to converge to some point near x. This effect can be avoided by periodically using Equation 10 to recompute the correct residual.

Before analyzing the convergence of Steepest Descent, I must digress to ensure that you have a solid understanding of eigenvectors.

5. Thinking with Eigenvectors and Eigenvalues

After my one course in linear algebra, I knew eigenvectors and eigenvalues like the back of my head. If your instructor was anything like mine, you recall solving problems involving eigendoohickeys, but you never *really* understood them. Unfortunately, without an intuitive grasp of them, CG won't make sense either. If you're already eigentalented, feel free to skip this section.

Eigenvectors are used primarily as an analysis tool; Steepest Descent and CG do not calculate the value of any eigenvectors as part of the algorithm[1].

5.1. Eigen do it if I try

An *eigenvector* v of a matrix B is a nonzero vector that does not rotate when B is applied to it (except perhaps to point in precisely the opposite direction). v may change length or reverse its direction, but it won't turn sideways. In other words, there is some scalar constant λ such that $Bv = \lambda v$. The value λ is an *eigenvalue* of B. For any constant α, the vector αv is also an eigenvector with eigenvalue λ, because $B(\alpha v) = \alpha Bv = \lambda \alpha v$. In other words, if you scale an eigenvector, it's still an eigenvector.

Why should you care? Iterative methods often depend on applying B to a vector over and over again. When B is repeatedly applied to an eigenvector v, one of two things can happen. If $|\lambda| < 1$, then $B^i v = \lambda^i v$ will vanish as i approaches infinity (Figure 9). If $|\lambda| > 1$, then $B^i v$ will grow to infinity (Figure 10). Each time B is applied, the vector grows or shrinks according to the value of $|\lambda|$.

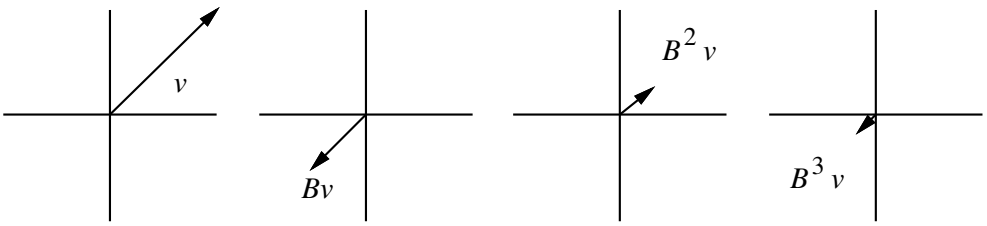


Figure 9: v is an eigenvector of B with a corresponding eigenvalue of -0.5. As i increases, $B^i v$ converges to zero.

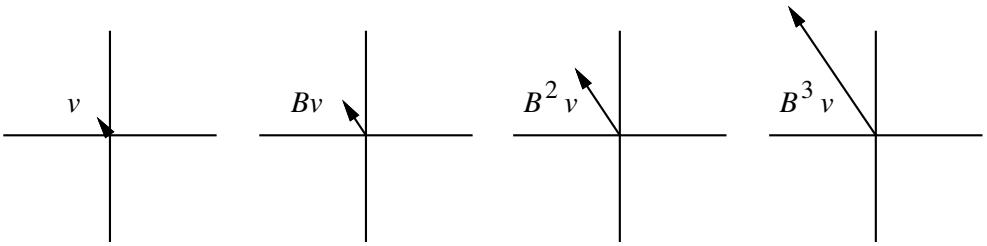


Figure 10: Here, v has a corresponding eigenvalue of 2. As i increases, $B^i v$ diverges to infinity.

[1]However, there are practical applications for eigenvectors. The eigenvectors of the stiffness matrix associated with a discretized structure of uniform density represent the natural modes of vibration of the structure being studied. For instance, the eigenvectors of the stiffness matrix associated with a one-dimensional uniformly-spaced mesh are sine waves, and to express vibrations as a linear combination of these eigenvectors is equivalent to performing a discrete Fourier transform.

If B is symmetric (and often if it is not), then there exists a set of n linearly independent eigenvectors of B, denoted $v_1, v_2, \ldots, v_n$. This set is not unique, because each eigenvector can be scaled by an arbitrary nonzero constant. Each eigenvector has a corresponding eigenvalue, denoted $\lambda_1, \lambda_2, \ldots, \lambda_n$. These are uniquely defined for a given matrix. The eigenvalues may or may not be equal to each other; for instance, the eigenvalues of the identity matrix I are all one, and every nonzero vector is an eigenvector of I.

What if B is applied to a vector that is not an eigenvector? A very important skill in understanding linear algebra — the skill this section is written to teach — is to think of a vector as a sum of other vectors whose behavior is understood. Consider that the set of eigenvectors $\{v_i\}$ forms a basis for $\mathbb{R}^n$ (because a symmetric B has n eigenvectors that are linearly independent). Any n-dimensional vector can be expressed as a linear combination of these eigenvectors, and because matrix-vector multiplication is distributive, one can examine the effect of B on each eigenvector separately.

In Figure 11, a vector x is illustrated as a sum of two eigenvectors v_1 and v_2. Applying B to x is equivalent to applying B to the eigenvectors, and summing the result. On repeated application, we have $B^i x = B^i v_1 + B^i v_2 = \lambda_1^i v_1 + \lambda_2^i v_2$. If the magnitudes of all the eigenvalues are smaller than one, $B^i x$ will converge to zero, because the eigenvectors that compose x converge to zero when B is repeatedly applied. If one of the eigenvalues has magnitude greater than one, x will diverge to infinity. This is why numerical analysts attach importance to the *spectral radius* of a matrix:

$$\rho(B) = \max |\lambda_i|, \qquad \lambda_i \text{ is an eigenvalue of } B.$$

If we want x to converge to zero quickly, $\rho(B)$ should be less than one, and preferably as small as possible.

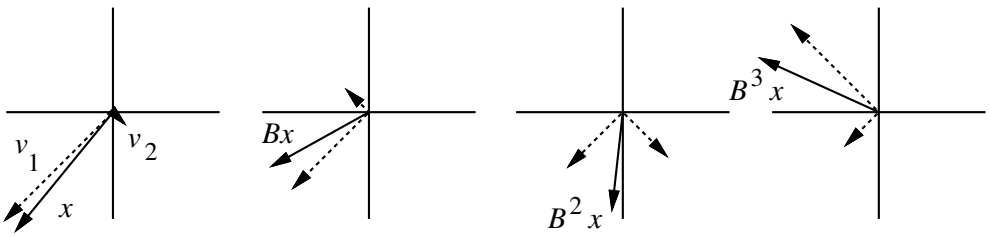


Figure 11: The vector x (solid arrow) can be expressed as a linear combination of eigenvectors (dashed arrows), whose associated eigenvalues are $\lambda_1 = 0.7$ and $\lambda_2 = -2$. The effect of repeatedly applying B to x is best understood by examining the effect of B on each eigenvector. When B is repeatedly applied, one eigenvector converges to zero while the other diverges; hence, $B^i x$ also diverges.

It's important to mention that there is a family of nonsymmetric matrices that do not have a full set of n independent eigenvectors. These matrices are known as *defective*, a name that betrays the well-deserved hostility they have earned from frustrated linear algebraists. The details are too complicated to describe in this article, but the behavior of defective matrices can be analyzed in terms of *generalized eigenvectors* and *generalized eigenvalues*. The rule that $B^i x$ converges to zero if and only if all the generalized eigenvalues have magnitude smaller than one still holds, but is harder to prove.

Here's a useful fact: the eigenvalues of a positive-definite matrix are all positive. This fact can be proven from the definition of eigenvalue:

$$\begin{aligned} Bv &= \lambda v \\ v^T B v &= \lambda v^T v. \end{aligned}$$

By the definition of positive-definite, the $v^T B v$ is positive (for nonzero v). Hence, λ must be positive also.

5.2. Jacobi iterations

Of course, a procedure that always converges to zero isn't going to help you attract friends. Consider a more useful procedure: the Jacobi Method for solving $Ax = b$. The matrix A is *split* into two parts: D, whose diagonal elements are identical to those of A, and whose off-diagonal elements are zero; and E, whose diagonal elements are zero, and whose off-diagonal elements are identical to those of A. Thus, $A = D + E$. We derive the Jacobi Method:

$$\begin{aligned} Ax &= b \\ Dx &= -Ex + b \\ x &= -D^{-1}Ex + D^{-1}b \\ x &= Bx + z, \qquad \text{where} \quad B = -D^{-1}E, \qquad z = D^{-1}b. \end{aligned} \tag{14}$$

Because D is diagonal, it is easy to invert. This identity can be converted into an iterative method by forming the recurrence

$$x_{(i+1)} = Bx_{(i)} + z. \tag{15}$$

Given a starting vector $x_{(0)}$, this formula generates a sequence of vectors. Our hope is that each successive vector will be closer to the solution x than the last. x is called a *stationary point* of Equation 15, because if $x_{(i)} = x$, then $x_{(i+1)}$ will also equal x.

Now, this derivation may seem quite arbitrary to you, and you're right. We could have formed any number of identities for x instead of Equation 14. In fact, simply by splitting A differently — that is, by choosing a different D and E — we could have derived the Gauss-Seidel method, or the method of Successive Over-Relaxation (SOR). Our hope is that we have chosen a splitting for which B has a small spectral radius. Here, I chose the Jacobi splitting arbitrarily for simplicity.

Suppose we start with some arbitrary vector $x_{(0)}$. For each iteration, we apply B to this vector, then add z to the result. What does each iteration do?

Again, apply the principle of thinking of a vector as a sum of other, well-understood vectors. Express each iterate $x_{(i)}$ as the sum of the exact solution x and the error term $e_{(i)}$. Then, Equation 15 becomes

$$\begin{aligned} x_{(i+1)} &= Bx_{(i)} + z \\ &= B(x + e_{(i)}) + z \\ &= Bx + z + Be_{(i)} \\ &= x + Be_{(i)} \qquad \text{(by Equation 14)}, \\ \therefore e_{(i+1)} &= Be_{(i)}. \end{aligned} \tag{16}$$

Each iteration does not affect the "correct part" of $x_{(i)}$ (because x is a stationary point); but each iteration does affect the error term. It is apparent from Equation 16 that if $\rho(B) < 1$, then the error term $e_{(i)}$ will converge to zero as i approaches infinity. Hence, the initial vector $x_{(0)}$ has no effect on the inevitable outcome!

Of course, the choice of $x_{(0)}$ does affect the number of iterations required to converge to x within a given tolerance. However, its effect is less important than that of the spectral radius $\rho(B)$, which determines the speed of convergence. Suppose that v_j is the eigenvector of B with the largest eigenvalue (so that $\rho(B) = \lambda_j$). If the initial error $e_{(0)}$, expressed as a linear combination of eigenvectors, includes a component in the direction of v_j, this component will be the slowest to converge.

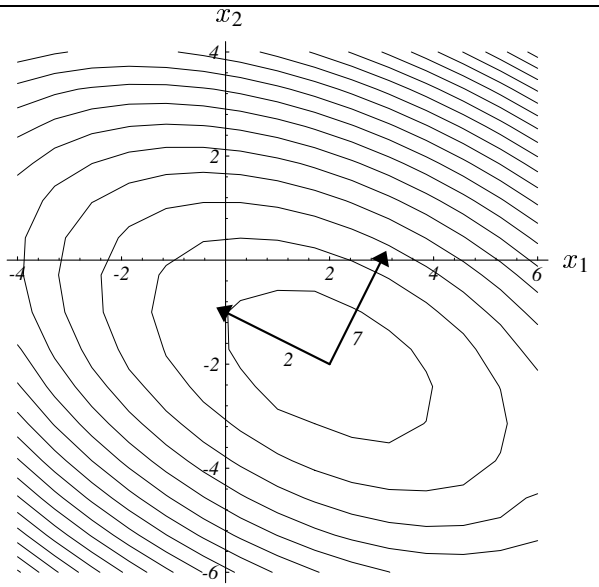


Figure 12: The eigenvectors of A are directed along the axes of the paraboloid defined by the quadratic form $f(x)$. Each eigenvector is labeled with its associated eigenvalue. Each eigenvalue is proportional to the steepness of the corresponding slope.

B is not generally symmetric (even if A is), and may even be defective. However, the rate of convergence of the Jacobi Method depends largely on $\rho(B)$, which depends on A. Unfortunately, Jacobi does not converge for every A, or even for every positive-definite A.

5.3. A Concrete Example

To demonstrate these ideas, I shall solve the system specified by Equation 4. First, we need a method of finding eigenvalues and eigenvectors. By definition, for any eigenvector v with eigenvalue λ,

$$Av = \lambda v = \lambda I v$$

$$(\lambda I - A)v = 0.$$

Eigenvectors are nonzero, so $\lambda I - A$ must be singular. Then,

$$\det(\lambda I - A) = 0.$$

The determinant of $\lambda I - A$ is called the *characteristic polynomial*. It is a degree n polynomial in λ whose roots are the set of eigenvalues. The characteristic polynomial of A (from Equation 4) is

$$\det \begin{bmatrix} \lambda - 3 & -2 \\ -2 & \lambda - 6 \end{bmatrix} = \lambda^2 - 9\lambda + 14 = (\lambda - 7)(\lambda - 2),$$

and the eigenvalues are 7 and 2. To find the eigenvector associated with $\lambda = 7$,

$$\begin{aligned} (\lambda I - A)v = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= 0 \\ \therefore 4v_1 - 2v_2 &= 0. \end{aligned}$$

Any solution to this equation is an eigenvector; say, $v = [1, 2]^T$. By the same method, we find that $[-2, 1]^T$ is an eigenvector corresponding to the eigenvalue 2. In Figure 12, we see that these eigenvectors coincide with the axes of the familiar ellipsoid, and that a larger eigenvalue corresponds to a steeper slope. (Negative eigenvalues indicate that f decreases along the axis, as in Figures 5(b) and 5(d).)

Now, let's see the Jacobi Method in action. Using the constants specified by Equation 4, we have

$$\begin{aligned} x_{(i+1)} &= -\begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{6} \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} x_{(i)} + \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{6} \end{bmatrix} \begin{bmatrix} 2 \\ -8 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -\frac{2}{3} \\ -\frac{1}{3} & 0 \end{bmatrix} x_{(i)} + \begin{bmatrix} \frac{2}{3} \\ -\frac{4}{3} \end{bmatrix} \end{aligned}$$

The eigenvectors of B are $[\sqrt{2}, 1]^T$ with eigenvalue $-\sqrt{2}/3$, and $[-\sqrt{2}, 1]^T$ with eigenvalue $\sqrt{2}/3$. These are graphed in Figure 13(a); note that they do not coincide with the eigenvectors of A, and are not related to the axes of the paraboloid.

Figure 13(b) shows the convergence of the Jacobi method. The mysterious path the algorithm takes can be understood by watching the eigenvector components of each successive error term (Figures 13(c), (d), and (e)). Figure 13(f) plots the eigenvector components as arrowheads. These are converging normally at the rate defined by their eigenvalues, as in Figure 11.

I hope that this section has convinced you that eigenvectors are useful tools, and not just bizarre torture devices inflicted on you by your professors for the pleasure of watching you suffer (although the latter is a nice fringe benefit).

6. Convergence Analysis of Steepest Descent

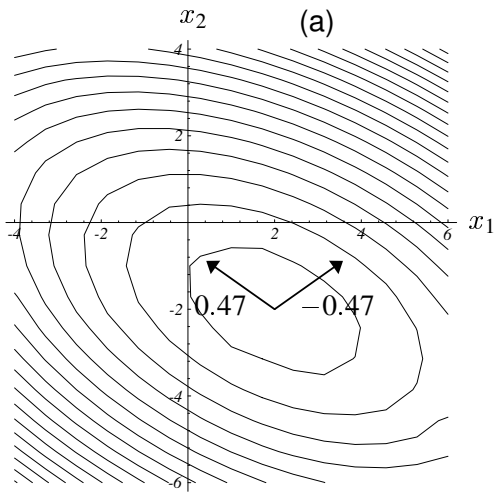
6.1. Instant Results

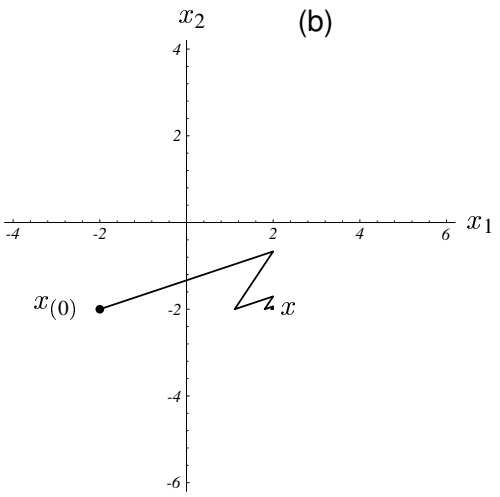
To understand the convergence of Steepest Descent, let's first consider the case where $e_{(i)}$ is an eigenvector with eigenvalue λ_e. Then, the residual $r_{(i)} = -Ae_{(i)} = -\lambda_e e_{(i)}$ is also an eigenvector. Equation 12 gives

$$\begin{aligned} e_{(i+1)} &= e_{(i)} + \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}} r_{(i)} \\ &= e_{(i)} + \frac{r_{(i)}^T r_{(i)}}{\lambda_e r_{(i)}^T r_{(i)}} (-\lambda_e e_{(i)}) \\ &= 0. \end{aligned}$$

Figure 14 demonstrates why it takes only one step to converge to the exact solution. The point $x_{(i)}$ lies on one of the axes of the ellipsoid, and so the residual points directly to the center of the ellipsoid. Choosing $\alpha_{(i)} = \lambda_e^{-1}$ gives us instant convergence.

For a more general analysis, we must express $e_{(i)}$ as a linear combination of eigenvectors, and we shall furthermore require these eigenvectors to be orthonormal. It is proven in Appendix C2 that if A is


x_2
(a)
x_1
0.47
-0.47


x_2
(b)
x_1
$x_{(0)}$
x